RUT - development handbook 13.1 How to find test cases for the system test v3.0-en

Mikael Blom

SUMMARY

When preparing the system test it can be difficult to find all test cases that are relevant. To minimize this problem some sort of method should be used. The method can give guidelines on how to find relevant test cases and in which context these test cases are important to perform. This document describes a process that helps the reader to find relevant test cases for the system test.

1 Field of Application

The process description gives a proposal to a number of different test types for the system test. It also describes what to look at to verify the importance of performing a certain test case. It can also be used as an aid when designing test cases. These test cases are written down in the system test specification and are used together with corresponding test script when performing the test.

2 Prerequisites

To be able to decide what should be tested some prerequisites must be fulfilled. When performing the system test one important test among several is to test the constructed system against the requirements specification. A prerequisite is that these requirements, both functional and non-functional, are well defined in the requirements specification.

The prerequisite when deciding about what should be in the resulting system test specification is that the requirements in the requirements specification are defined in a way so that they are testable. Otherwise it is easy to end up in a situation when there is a problem knowing if a certain test case passed or failed after the test has been performed.

A testable requirement is a requirement that has been defined in a way so that a test case can be constructed and the given result when performing the test decides if the requirement is fulfilled or not fulfilled. The result must be easy to verify. An example is "The maximum search time is 5 seconds" or "An error message should be shown when pressing button X and no in data has been written in the Y field". These requirements are testable, the first since the search time can be measured in seconds, and the second because an error message is shown or it is not shown, there is no other alternative.

An example on a requirement that is not testable is "The graphical user interface should be nice-looking" because it can not be measured without bias.

The requirement specification must also be complete so that all requirements are defined.

3 Realization

3.1 Overview

To decide what are relevant to test on the constructed system the first thing is to test it against the requirements specified by the customer. Then the system could also be tested against the users manual and the technical documentation.

At last there are some standard points that can be of general interest. For each of these standard points a decision has to be made to decide the importance for perform that kind of test on the system. In figure 1 an illustration is made about the above mentioned sources for construction of the system test. The result is documented in the system test specification.

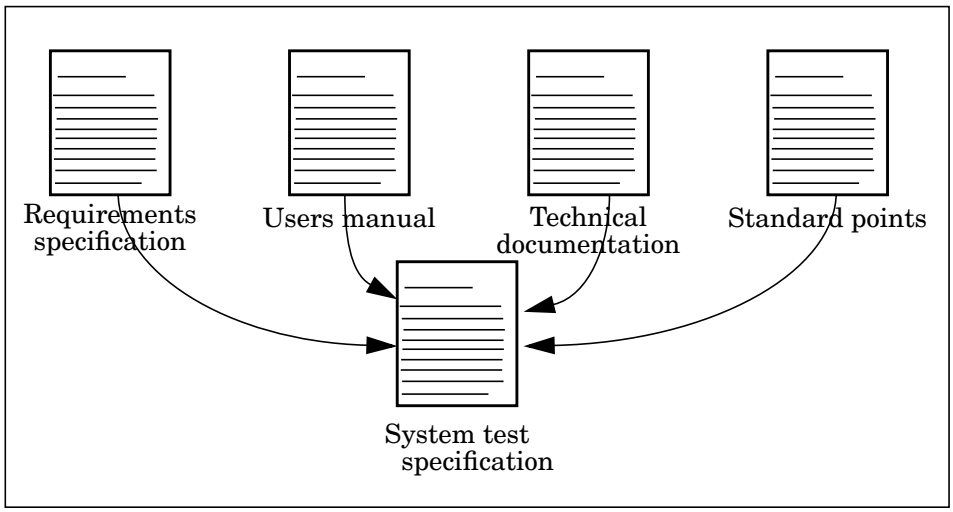

Figure 1. Sources when creating test cases for the system test.

3.2 Detailed description

What needs a more detailed description is the standard points mentioned above. These standard points are listed below and each will be given a short explanation.

- Stress test

  When performing stress test, the system is tested to work correctly during high load. This is important when for example many users simultaneously works with the system or a lot of parallel transactions are performed. It can also be good to see how the system responds when there is competition for resources. This kind of test is most useful when a multi user system is tested. This test is often done together with volume and efficiency test.

- Volume test

Volume test is when checking how the system responds when it is loaded with a large amount of data. Important things are to verify are that no data are lost and that the system does not freeze or fail because of too small internal buffers etc. This test is often done together with stress and efficiency test.

- Correct environment

  "The only way to prove that a product functions correctly is to use it in the real world on real data" (Daniel J. Mosley 1993).
  A system should be tested to such an extent as possible in the environment it is developed for. This concerns operating systems, hardware, real data etc. This eliminates the risk for failures when transferring the system from the develop environment to the target environment.

- Compatibility with earlier systems

  If the constructed system is a further development of an already existing system there may exist requirements specifying that some things should have equal function as in the earlier system. This often concerns graphical user interfaces when the users for example wants old functions to be called with the same function keys. This should be a part of the system test.

- Security

  What happens if there is a power failure, or if the user gives the system wrong input or a node in a distributed system fails? The system should be tested to things that normally not happen. This to verify that no serious damages or something unexpected happen to the system.

- Memory consumption

  It could be important to investigate the memory consumption of the system e.g. if the target machine has low memory capacity or to see that no unnecessary memory consumption is used.

- Efficiency

  Here the response time could be tested. To make it more critical the test could be done in parallel with the stress test and volume test.

- Installation instruction

  The system installation should be tested against the installation instructions to verify that the system without any problems can be installed by the customer.

- Reliability in operation

  This should be tested if there are any requirements specifying that there can not be any errors in the system. This concerns for example monitoring systems or systems where the customer loses a large amount of money if an failure occurs.

- Restart

If the system need to be restarted, because of some error, so can it in some systems be important to test that no information is lost. This is for example important in banking systems. It could also be important to test how long time it takes to restart the system, this to know how long the time the system services not are available.

- Service functions in the system

  It is also important to check that functions that are used for diagnosing errors really finds the correct failure and give as straight answer as possible.

- Inspection of documentation

  Manuals and other documentation that will be read by the customer should be inspected so that they are understandable. This also applies to documentation that will be read by those who will maintain the system.

- Human machine interaction

  If the system has a user interface this should be tested. The interface should be, depending on the customer, easy to use and any error- or informal messages should be understandable for the user.

- Procedure test

  Most computer systems are a part of a larger system mixed with human procedures. There should be some testing of the procedures that are part of the human system interacting with the computer system.

- Customer required test

  At last something should be produced with the system that shows that the system works as the customer required. These test scenarios could be designed in the definition phase together with the customer.

4 Result

4.1 Products

The above mentioned results in a system test specification where all the produced test cases are defined and documented. These should be grouped in the order of areas that they will test.

4.2 Product templates

Here follows an example on how a test case can be documented in the system test specification.

Administrator: Search for a contact

This test case will point out that an administrator can search for an administrator.

Tracability

Requirement R14, R23, R24, R25.

Prerequisites

The startpage of the system is shown.

Realization

1 Search for all contacts with the name Anders Persson.
2 Search for all contacts with the telephone number that starts with 0341.
3 Search for all contacts with the telephone number that starts with 0342 and is an employee at the company Nisses AB.

Expected result

1 Two contacts with the name Andres Persson is shown, one with the telephone number 0340-00 00 00 and the other one with 0340-00 00 01.
2 The contacts Orvar Bertilsson, Bengt Sändh and Ceasar Karlsson is shown.
3 The contact Adam Green and Otto Zetterlund is shown.

Figure 2. An example on how a test case for system test can be documented.

- Name

  At the top of figure 2 the name of the test case is given. The name uniquely defines the test case. Then follows a short description of what will be tested by the test case.

- Traceability

  Under this headline a reference to the origin of the test case is given. This could for example be a reference to one or more requirements specified by the customer that the test case should fulfil. It could also be a reference to a functionality, user interface or scenario described in the design document, or a reference to the user manual or one or more of the standard points mentioned in chapter 3.2.

- Prerequisites

This headline specifies the prerequisites that needs to be fulfilled, for the test to be viable.

- Realization

  The realization headline specifies step by step instructions how the test case will be accomplished. At last, the expected result headline gives a description about what is the correct result in each step.

5 Templates and forms

Here is a checklist that could be used to point out which test case types that are relevant to perform during the system test. Those tests that will be part of the system test are marked in the "Should be tested"-column. Then the "Finished"-column is marked when a sufficient number of test cases for that particular test case type have been designed.

Table 1: A checklist for different test case types at the system test

Type of test case	Should be tested	Finished
All requirements in the requirements specification are tested		
Test cases for the users manual		
Test cases for the technical documentation		
Stress tests		
Volume tests		
Tests in their correct environment		
Compatibility tests		
Security tests		
Memory tests		
Efficiency tests		
Test cases for the installation instructions		
Reliability tests		
Restart tests		
Service function tests		
Inspection of the documentation		
Test cases for the user interface		
Procedure tests		
Customer required test		

6 Verification of results

To perform a control of the system test specification it first has to be compared with the requirements specification and verify that for all requirements, both functional and non-functional, one or several test cases have been defined. It could also be compared with the user manual. Then at last it could be compared with the above mentioned standard points and make sure that those points considered relevant for system testing are specified in the test specification.

7 Examples with explanations

An example is here given from the PUM-course. The system which was constructed was a web calender where users should be able to upload activities. These activities can then be viewed by other users. Most of the above described standard points are already requirements in the requirements specification, anyway they will one and one be brought up to discussion in this example. To start with we need to add the test cases for the requirements in the requirement specification to the test specification. Test cases for correct feedback to the user should also be added. After that you could look at the standard points and see which of those that are necessary to add.

- Stress test is performed to verify that multiple users can gain access to the calender at the same time. If large amount of users tries to connect to the server and are refused, users will be irritated and perhaps find another calender somewhere else.
- Volume test is also relevant to see how the system reacts when a large amount of activities are stored in the database. This hopefully verifies that the database structure is good. Long search time will be annoying for the users.
- Correct environment should be tested to make sure that the system is able to perform well where it is supposed to.
- The compatibility can not be tested since there is no earlier versions of the calender.
- The security needs to be tested carefully. It is important that users shouldn't be able to bring the database down just by typing some sql like commands when creating an activity. Since you can list activities by date after a search, test that verifies that a user cannot enter a date of an invalid type should be performed. Input from user to the system should be checked to make sure that it doesn't contain any HTML tags so the layout could be damaged.
- Since there is no specification requirements on memory consumption its a good idea to test it, e.g. the system may be running on an old machine.
- The efficiency can not be low because then the users will find some faster calender they will use instead. Same problem that occurs on stress och volume test.
- Installation instruction should be tested to verify that the system can be installed by the customer.

- The reliability in operation must also be tested so that the calender does not go down very often. This can be performed by having the calender up and running for some time to see how it performs.
- Restart test verifies that no data is lost due to the restart.
- Service functions are not implemented in such a small system and therefore cannot be tested.
- The documentation must be verified so that the people who are maintaining the system can carry out their work fast and simple. It is also important if the system is to be further expanded by people who haven't wrote the source code.
- Human machine interaction is important to test to see if future users finds the application attractive. These tests gives information about whether the grouping of the calender information is logical. Here you also verify that the feedback given to the user is correct.
- At the end customer requirement should be tested, which in this case is that users should be able to create and view activities.

8 Solutions to common problems

- I don't know were to start.

  Sometimes it can be hard to know were to begin, if that happen it's a good idea to start with the requirements specified by the customer. More about this can be found in section 3.1 on page 4.

- There is not possible to write meaningful test cases for the requirements.

  The testability has not been taken in to consideration when defining the requirements in the requirement specification. The only thing that can be done is to renegotiate the requirements so that they will be less diffuse. For an example can "The user interface should look nice" be reformulated to "Of 10 random selected persons in the age of 15 - 20 years should at least eight of them think that the user interface looks nice".

- There is not enough time to perform all test cases.

  Because the system test is carried out late in the project there is a great risk that the tests are delayed because of earlier delays in the project. This makes it important to prioritize the test cases and carry out the most important ones in the beginning of the test phase. That is, the test cases that will have largest impact on finding error in the system should be tested first. You should make a test order when specifying the test cases.

- There is to many test cases.

  As a groundrule it can't be to many test cases. The only boundary is time and if that occur see question "There is not enough time to perform all test cases."

- The implementation take to much time.

  If the implementation phase isn't complete, some restructuring may be in order. Start by testing the parts that is complete. This isn't efficient and a better way is to make the parts be ready in the order that is optimal for

testing. As a last resource see question "There is not enough time to perform all test cases."

- How do I know that the tests test the complete system?

  This is a very difficult question and practical impossible to have a concrete answer to. If you have tested it for all requirements specified by the customer, the users manual, the technical documentation and checked the standard points, the system is tested for the requirements the customer have. However this doesn't mean that the complete system is tested, but if you should test the complete system for all possible mistake, it can be tested infinitely.

- I do not know anything about programming.

  You doesn't need to have much programming experience to write the test cases, but it makes it easier. Much is common sense, this because the test cases is written in plain text.
  However for the testskript that is build on the test cases, programming experience is necessary.

- The equipment is troublesome and we can not perform the tests in their correct environment.

  Hardware is very unreliable and to perform tests in their correct environment they must be planed a long time before their realization. Particularly when the customer places the equipment at their disposal. Having regular and good contact with the customer and/or the course administration makes it easier to get what you want in time.

9 Adjustment to the PUM-course

In the end of the PUM-course a system test shall be performed. This is carried out simplest by first creating test cases that test all requirements. Then if not already defined the test cases for the users manual and technical documentation should be created. If there is any time left the standard points described in chapter 3.2 could be evaluated. Most of these standard points are unnecessary and impossible to carry out in the time frame of the PUM-course. How much work you want to spend on the test work is up to the whole group to decide.

10 Measurement of the process

10.1 Resource measurement

It is probably very difficult to formulate a measurement for evaluation of the resources (time, personnel, aids etc) that is needed for finding and documenting test cases for the system test. Since this on the whole never is mentioned in the examined literature and no attempts to define such a measurement has been made, nothing more can be said here.

10.2 Product measurement

Neither has a measurement for evaluation of the test cases created with this process been found.

10.3 Forms for collection of data

Those forms that is needed for collection of measurement data over the measurements described in chapter 10.1 Resource measurement and chapter 10.2 Product measurement depends on the these measurements and can thus not be created because no measurements are available.

11 History of the process

Table 2: History of the process

Version	Date	Comment	Author
1. 0	940217	The first version as a part of a home exam, PUM-course -94.	Patrick Lindholm
1. 1	960206	A first check of the document before the home exam in the PUM-course -96.	Nicklas Hjalmarsson
1. 2	960214	Improvements made as a part of a home exam, PUM-course -96. For more information see 13.3 internal comments.	Nicklas Hjalmarsson
1. 3	980527	Improvements made as a part of a home exam, PUM-course 98. For more information see 13.3 internal comments.	Anders Bergsten
1. 4	000611	Improvements of the process was made as a part of the PUM-course -99/00.	Martin Öberg
1.4-en	030631	Document translated into english. Part of the PUM-course -03.	Mikael Löfqvist

Version	Date	Comment	Author
2.0-en	050113	Chapter 7 rewritten. Grammar is corrected.	Karl Andersson
3.0-en	060117	Improvements on chapter 7 and 8. Grammar is corrected.	Mikael Blom

12 Changes not yet attended to

A more thorough investigation needs to find out if there exists a resource measurement for measuring the need of resources that could be suitable to carry out the process. That is to find and document test cases for the system test.

Furthermore needs a thorough investigation to find out if there exists a product measurement for measuring the result of the process. That is to measure the result of the found test cases.

There should be more solutions to common problems. That is what you want to read about in a RUT, not forms.

The RUT may be should be enlarged with information about the realization of the system test.

13 References

13.1 Method of description

Own experiences from Ericsson Telecom AB and Ericsson Radio Systems AB

Own experiences (Anders Bergsten) as responsible for test in the PUM-course

Lecture notes by Christian Krysander

Glenford, J. Myers (1976) Software Relability

Mosley, Daniel J. (1993) The handbook of MIS application software testing

13.2 Method evaluation

Nothing has been done that can be evaluated.

13.3 Internal comments

Patrik Lindholm: *Since there is not so much mentioned about system test and that I have not found any direct method in the books that I have read the method described above is one of the best sources that I have mentioned under references.*

Nicklas Hjalmarsson: Neither I have found any material describing system test, most books only mentions it but no thorough explanation is given. The only book that I have found that describes the system test is Mayers book, I have also tried to search on the Internet for information, but it resulted only in homepages of other PUM-groups. My changes to this document is therefore small and is mostly concerning cleaning up the language and formulation, adding of new points and also some changes in the detailed description.

Anders Bergsten: I can only agree with Patrik Lindholm and Nicklas Hjalmarsson. This RUT is very good and the most changes I have done are to increase the understanding and readability of the document. I have also in chapter 4.2 Product templates added a product template describing how a test case could be documented. Furthermore have I added an description of a table (see Table 2), and reformulated it so that it hopefully will be more useful.